|  |
| --- |
| **DAY 18 ASSIGNMENT**  **By**  **ARUN KUMAR YADLAPALLI**  **@**  **NB Healthcare Technologies PVT LTD.** |

|  |
| --- |
| **Q1) What is the use of XML** |
| **A)**   * XML stands for Extensible Markup Language. * XML is used for Universal data transfer mechanism to send data across different platforms. |

|  |
| --- |
| **Q2) Write the points discussed about xml in the class** |
| **A)**   * XML has user defined Tags. * XML is Case Sensitive. * In XML there should be only one Root Tag. * XML is used for Universal data transfer mechanism to send data across different platforms. * There are 2 types in XML – Tag Based XML (Takes more Size)   - Attribute Based XML. (Takes less size)   * XML is not Platform dependent. It’s just a text file. |

|  |
| --- |
| **Q3) Create a simple xml to illustrate: a. Tag based xml with 10 products b. Attribute based xml** |
| **Code:**  **Tag based XML:**  <Products>  <Table>  <Type>wood</Type>  <madeby>carpenter</madeby>  </Table>    <Tv>  <company>samsung</company>  <mftyear>2019</mftyear>  </Tv>  <Tv>  <company>hitachi</company>  <mftyear>2017</mftyear>  </Tv>  <Tv>  <company>mi</company>  <mftyear>2020</mftyear>  </Tv>  <Tv>  <company>lg</company>  <mftyear>2015</mftyear>  </Tv>  <monitor>  <company>dell</company>  </monitor>  <monitor>  <company>acer</company>  </monitor>  <monitor>  <company>zebronics</company>  </monitor>  <monitor>  <company>intel</company>  </monitor>  <monitor>  <company>Lenovo</company>  </monitor>      </Products>  **Attribute based XML:**  <Products>    <table type="wood" madeby="carpenter" />    <tv company="samsung" mftyear="2019"/>    <tv company="hitachi" mftyear="2017"/>    <tv company="mi" mftyear="2020"/>    <tv company="lg" mftyear="2015"/>    <monitor company="dell"/>    <monitor company="acer"/>    <monitor company="zebronics"/>    <monitor company="intel"/>    <monitor company="lenovo"/>    </Products> |
| **Tag based XML:**    **Attribute based XML:** |

|  |
| --- |
| **Q4) Convert the above xml to JSON and display the JSON data** |
| **XML Data:**  <Products>  <Table>  <Type>wood</Type>  <madeby>carpenter</madeby>  </Table>    <Tv>  <company>samsung</company>  <mftyear>2019</mftyear>  </Tv>  <Tv>  <company>hitachi</company>  <mftyear>2017</mftyear>  </Tv>  <Tv>  <company>mi</company>  <mftyear>2020</mftyear>  </Tv>  <Tv>  <company>lg</company>  <mftyear>2015</mftyear>  </Tv>  <monitor>  <company>dell</company>  </monitor>  <monitor>  <company>acer</company>  </monitor>  <monitor>  <company>zebronics</company>  </monitor>  <monitor>  <company>intel</company>  </monitor>  <monitor>  <company>Lenovo</company>  </monitor>      </Products> |
| **Converting to JSON:**  {  "Products": {  "Table": {  "Type": "wood",  "madeby": "carpenter"  },  "Tv": [  {  "company": "samsung",  "mftyear": 2019  },  {  "company": "hitachi",  "mftyear": 2017  },  {  "company": "mi",  "mftyear": 2020  },  {  "company": "lg",  "mftyear": 2015  }  ],  "monitor": [  {  "company": "dell"  },  {  "company": "acer"  },  {  "company": "zebronics"  },  {  "company": "intel"  },  {  "company": "Lenovo"  }  ]  }  } |

|  |
| --- |
| **Q5) Research and write the benefits of JSON over XML** |
| **A)**   * In most scenarios, JSON is undoubtedly easier to read in its expanded form than XML. * JSON can have a substantially lower character count reducing the overhead in data transfers. * JSON is much easier to parse. |

|  |
| --- |
| **Q6) For the below requirement, create a layered architecture project with seperate class library for Business logic. create console application create windows(or desktop) application Business Requirement: FIND FACTORIAL OF A NUMBER: 0 = 1 positive number (upto 7) = factorial answer > 7 = -999 (as answer) < 0 = -9999 (as answer)** |
| **Code:**  namespace ArunKLibrary  {  public class Algebra    {  public static int Factorial(int n)  {  if (n == 0)  return 1;  else if (n > 7)  return -999;  else if (n < 0)  return -9999;  else    {  int fact = 1;  for (int i = 1; i <= n; i++)  {  fact = fact \* i;    }  return fact;  }    }  }  }  namespace Consoleappp  {  class Program  {  static void Main(string[] args)  {  Console.WriteLine(Algebra.Factorial (-4));  Console.ReadLine();  }  }  }  namespace WindowsFormsApp  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }    private void button1\_Click(object sender, EventArgs e)  {  int input = int.Parse (textBox1.Text);  int fact = Algebra.Factorial(input);  textBox2.Text = fact.ToString();      }  }  } |
| **Output:** |

|  |
| --- |
| **Q7) For the above method, Implement TDD and write 4 test cases and put the code in word document. put the screen shot of all test cases failing. make the test cases pass.** |
| **Code:**  namespace ArunKLibrary  {  public class Algebra    {  public static int Factorial(int n)  {  if (n == 0)  return 1;  else if (n > 7)  return -999;  else if (n < 0)  return -9999;  else    {  int fact = 1;  for (int i = 1; i <= n; i++)  {  fact = fact \* i;    }  return fact;  }  }    }  }  namespace ArunKLibrary.Tests  {  [TestClass()]  public class AlgebraTests  {  [TestMethod()]  public void FactorialTest\_0\_input()  {  //Arrange  int input=0;  int expected = 1;      //Act  int actual = Algebra.Factorial(input);  //Assert  Assert.AreEqual(expected, actual);  }    [TestMethod()]  public void FactorialTest\_Greaterthan7\_input()  {  //Arrange  int input = 8;  int expected = -999;      //Act  int actual = Algebra.Factorial(input);  //Assert  Assert.AreEqual(expected, actual);  }    [TestMethod()]  public void FactorialTest\_Lessthan0\_input()  {  //Arrange  int input = -6;  int expected = -9999;      //Act  int actual = Algebra.Factorial(input);  //Assert  Assert.AreEqual(expected, actual);  }    [TestMethod()]  public void FactorialTest\_Fact\_input()  {  //Arrange  int input = 5;  int expected = 120;      //Act  int actual = Algebra.Factorial(input);  //Assert  Assert.AreEqual(expected, actual);  }  }  } |
| **Output:** |

|  |
| --- |
| **Q8) Add one more method to check if the number is palindrome or not in the above Algebra class and write test case for the same.** |
| **Code:**  public static string Palindrome(int n)  {  int sum = 0;  int r;  int temp = n;  while (n > 0)  {  r = n % 10;  sum = sum \* 10 + r;  n = n / 10;  }  if(temp==sum)  return "Palindrome";  else  return "Not a Palindrome";    }  public void PalindromeTest()  {  //Arrange  int n = 121;  string expected = "Palindrome";  //Act  string actual = Algebra.Palindrome(n);  //Assert  Assert.AreEqual(expected, actual);  }  [TestMethod()]  public void PalindromeTest\_Wrong()  {  //Arrange  int n = 443;  string expected = "Not a Palindrome";  //Act  string actual = Algebra.Palindrome(n);  //Assert  Assert.AreEqual(expected, actual);  } |
| **Output:** |